

# TIMED AUTOMATA

## AN OVERVIEW

KWAME OKYERE OWUSU-BOAKYE  
VARUN YADAV KESHABOINA

APRIL 6, 2020



# TABLE OF CONTENTS

- 1 Introduction
- 2 Aim and Objectives
- 3 Literature Review
- 4 Discussions
  - Transition Systems
    - Complex Transition Systems
    - Transition Systems With Timing Constraints
  - Timed Automata
    - Clock Valuation
    - Run
    - Complex Timed Automata
  - Reachability Analysis
    - Time Abstract Transition Systems
  - Decidability issues
  - UPPAAL Implementation
  - Limitations of Timed Automata
- 5 Conclusion

# INTRODUCTION

# INTRODUCTION

## Abstract

Due to how integral the absence of undesirable behaviour of real-time critical systems is to ensure safety, **Model checking** aims at formally verifying a model of the system against a **correctness** property.

## Timed Automata

**Timed automata (TAs)** are a popular formalism to model and verify safety critical systems with timing constraints. **TAs** extend **Finite State Automata (FSA)** with clocks, i. e., real-valued variables increasing linearly

**TAs** provides a simple way to annotate state-transition graphs with timing constraints using finitely many real-valued clock variables and relies on the construction of a finite quotient of the infinite space of clock valuations. [1]

**Timed Automata** benefit from many interesting decidable properties, such as the emptiness of the accepted language, the reachability of a control state.

Other problems are undecidable though, such as the universality of the accepted timed language; in addition, given a TA, building a TA recognizing the complement of the timed language of the first TA cannot be achieved in general.

TAs were also studied in a robust version, i. e., when all timing guards can be enlarged or shrunk by an infinitesimal constant factor, without changing the language, the reachability of a control state, etc

# **AIM AND OBJECTIVES**

# AIM AND OBJECTIVES

The aim of this project is to discuss time-constrained finite state systems and learn about;

- The difference between transition systems and time constrained transition systems
- Clock constraints and clock interpretations
- Reachability analysis
- Decidability analysis
- Limitations of Timed automata

# LITERATURE REVIEW



# LITERATURE REVIEW

- In 1994, Rajeev Alur and David L. Dill introduced *A Theory of Timed Automata*.
- Béatrice Bérard, Volker Diekert, Paul Gastin and Antoine Petit followed up with *Characterization of the expressive power of silent transitions in timed automata* in 1998
- Béatrice Bérard and Catherine Dufourd talked about *Timed automata and additive clock constraints* in 2000
- Decidability issues were brought up in 1997 by Catherine Dufourd.
- In 2008, the concept of alternating timed automata was brought up by Pawel Parys and Igor Walukiewicz

# DISCUSSIONS

# TRANSITION SYSTEMS

A transition system  $\mathbf{S}$  is a tuple  $\langle Q, q_{start}, \Sigma, \rightarrow \rangle$  consisting of

- A set of states  $Q$
- An initial state  $q_{start} \in Q$
- A set of labels or events  $\Sigma$
- A set of transitions  $\rightarrow \subseteq Q \times \Sigma \times Q$

The system starts from an initial state,  $q_{start}$  and if  $q \xrightarrow{a} q'$  then the system can change its state from  $q$  to  $q'$  on event  $a$

The state  $q'$  is reachable from the state  $q$  if  $q \rightarrow^* q'$

The state  $q$  is a reachable state of the system if  $q$  is reachable from some initial state

# COMPLEX TRANSITION SYSTEMS

## Complex System

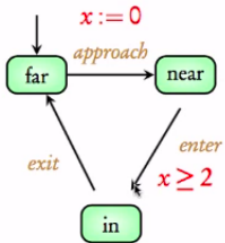
This can be described as a product of interacting transition systems

Let,  $S_1 = \langle Q_1, q_{1start}, \Sigma_1, \rightarrow_1 \rangle$  and  $S_2 = \langle Q_2, q_{2start}, \Sigma_2, \rightarrow_2 \rangle$  be two transition systems

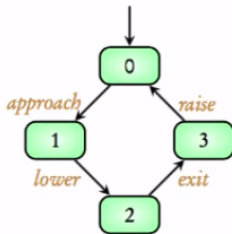
The product, denoted as  $S_1 \parallel S_2 = \langle Q_1 \times Q_2, q_{1start} \times q_{2start}, \Sigma_1 \cup \Sigma_2, \rightarrow \rangle$

$(q_1, q_2) \xrightarrow{a} (q'_1, q'_2)$  iff either

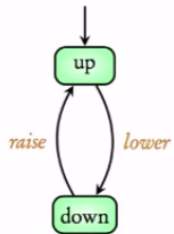
- $a \in \Sigma_1 \cap \Sigma_2$  and  $q_1 \xrightarrow{a}_1 q'_1$  and  $q_2 \xrightarrow{a}_2 q'_2$
- $a \in \Sigma_1 - \Sigma_2$  and  $q_1 \xrightarrow{a}_1 q'_1$  and  $q_2 = q'_2$
- $a \in \Sigma_2 - \Sigma_1$  and  $q_1 = q'_1$  and  $q_2 \xrightarrow{a}_2 q'_2$



Train



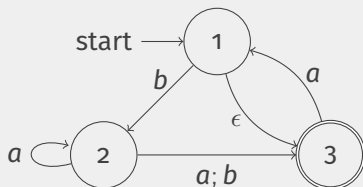
Controller



Gate

# TRANSITION SYSTEMS WITH TIMING CONSTRAINTS

To equip a transition system with timing constraints, we consider it as a finite graph with a finite set of real-valued clocks.

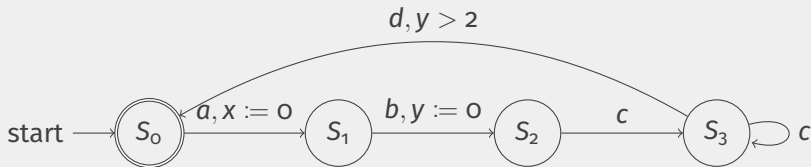


The vertices of the graph are called **locations**, and edges(transitions) are called **switches**

While switches are instantaneous, time can elapse in a location

# CLOCK CONSTRAINTS AND INTERPRETATION

- A clock can be reset to zero simultaneously with any switch.
- At any instant, the reading of a clock equals the time elapsed since the last time it was reset.
- With each switch we associate a clock constraint, and require that the switch may be taken only if the current values of the clocks satisfy this constraint.
- With each location we associate a clock constraint called its **invariant**, and require that time can elapse in a location only as long as its invariant stays true.



# TIMED AUTOMATA

A timed automaton is a tuple  $\mathcal{A} = \langle L, L_{start}, \Sigma, X, I, F, E \rangle$  that consists of the following components:

- A finite set of locations,  $L$
- An initial location,  $L_{start} \subseteq L$
- A finite set called the alphabet or actions of  $\mathcal{A}$ ,  $\Sigma$
- A finite set of clocks,  $X$
- A mapping that labels each location  $s$  with some clock constraint in  $\mathcal{P}(X)$  called the **Invariant**,  $I$ .
- A set of accepting locations,  $F \subseteq L$
- A set of transitions/edges/switches,  $E \subseteq L \times \Sigma \times \mathcal{B}(X) \times \mathcal{P}(X) \times L$ 
  - ▶  $\mathcal{B}(X)$  is the set of clock constraints involving clocks from  $X$
  - ▶  $\mathcal{P}(X)$  is the powerset of  $X$ .

An edge  $(\ell, a, g, r, \ell')$  from  $E$  is a transition from locations  $\ell$  to  $\ell'$  with action  $a$ , guard  $g$  and clock resets  $r$ .



## Clock Constraint

For a set  $\mathbf{X}$  of clocks, the set  $\mathcal{P}(\mathbf{X})$  of clock constraints  $\varphi$  is defined by the grammar

$$\varphi := x \leq c \mid c \leq x \mid x < c \mid c < x \mid \varphi_1 \wedge \varphi_2$$

where  $\mathbf{x}$  is a clock in  $\mathbf{X}$  and  $\mathbf{c}$  is a constant in  $\mathbb{N}$ , example:  $\mathbf{x} \leq 5 \wedge \mathbf{x} \geq 3$

A **clock interpretation** or **clock valuation**,  $\mathbf{v}$  for a set of clock  $\mathbf{X}$  is a function  $\mathbf{v} : \mathbf{X} \rightarrow \mathbb{R}^+$  that assigns to each clock  $\mathbf{x} \in \mathbf{X}$  its current value  $\mathbf{v}(\mathbf{x})$ .

For  $\delta \in \mathbb{R}$ ,  $\mathbf{v} + \delta$  denotes the clock interpretation which maps every clock  $x$  to the value  $\mathbf{v}(x) + \delta$ . For  $Y \subseteq X$ ,  $\mathbf{v}[Y := \mathbf{o}]$  denotes the clock interpretation for  $\mathbf{X}$  which assigns  $\mathbf{o}$  to each  $x \in Y$ , and agrees with  $\mathbf{v}$  over the rest of the clocks, i.e satisfies the **invariant**.

## Timed Word

This is an extension of the notion of words in **formal language** where each letter is associated with a positive **time tag** in a non-decreasing order.

Given an alphabet  $A$ , a **timed word** is a sequence, finite or infinite  $w = (a_0, t_0)(a_1, t_1) \dots$  with  $a_i \in A$ ,  $t_i \in \mathbb{R}^+$  with  $t_i \leq t_{i+1}$  for each  $i$ .

Given a timed word  $w = (a_0, t_0)(a_1, t_1) \dots$  a **run** is a sequence of the form  $(\ell_0, \nu_0) \xrightarrow[t_1]{a_1} (\ell_1, \nu_1) \dots$  satisfying the following conditions.

- initialization,  $\ell_0 \in L_0$
- consecution, for all  $i \geq 1$

## Consecution

There exists an edge in  $E$  of the form  $\langle \ell_{i-1}, a_i, g_i, r_i, \ell_i \rangle$  such that:

- we assume that  $t_i - t_{i-1}$  time units passed, and at this time, the guard is satisfied, i.e.  $\nu_{i-1} + t_i - t_{i-1}$  satisfies  $g_i$ ,
- the new clock valuation  $\nu_i$  corresponds to  $\nu_{i-1}$ , in which  $t_i - t_{i-1}$  time units passed and in which the clocks of  $r_i$  where reset. Formally,  $\nu_i = (\nu_{i-1} + t_i - t_{i-1})[r_i \rightarrow 0]$

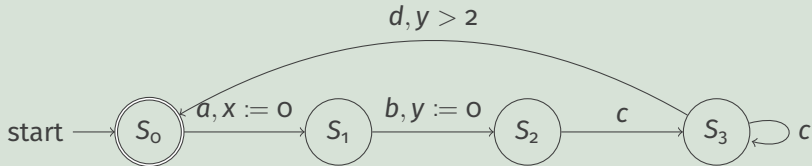
For a timed automaton  $\mathcal{A}$ , associated with a transition system  $S_A$ , a state of  $S_A$  is a pair  $(\mathbf{s}, \mathbf{v})$  such that  $\mathbf{s}$  is a location of  $\mathbf{A}$  and  $\mathbf{v}$  is a clock valuation for the set of clock,  $\mathbf{X}$  such that  $\mathbf{v}$  satisfies the invariant  $\mathbf{I}(\mathbf{s})$ .

A state  $(\mathbf{s}, \mathbf{v})$  is an initial state if  $\mathbf{s}$  is an initial location of  $\mathbf{A}$  and  $\mathbf{v}(\mathbf{x}) = \mathbf{0}$  for all clocks  $\mathbf{x}$ .

There are two types of transitions in  $S_A$ :

- **Elapse of time:** for a state  $(s, v)$  and a real-valued time increment  $\delta \geq 0$ ,  $(s, v) \xrightarrow{\delta} (s, v + \delta)$  if for all  $0 \leq \delta' \leq \delta$ ,  $v + \delta'$  satisfies the invariant  $I(s)$ . This transition is called a **delay transition**.
- **Location switch:** for a state  $(s, v)$  and a switch  $\langle s, a, \varphi, \lambda, s' \rangle$  such that  $v$  satisfies  $\varphi$ ,  $(s, v) \xrightarrow{a} (s', v[\lambda := 0])$ . Such a transition is called a **discrete transition**.

## Example



►  $(S_0, 0, 0) \xrightarrow{1.2} (S_0, 1.2, 1.2) \xrightarrow{a} (S_1, 0, 1.2) \xrightarrow{0.7} (S_1, 0.7, 1.9) \xrightarrow{b} (S_2, 0.7, 0)$

# PROPERTIES OF TIMED TRANSITION SYSTEMS

## Null delay property

It is always possible to delay for 0 time units.

$$(s, v) \xrightarrow{0} (s, v)$$

## Time-additivity property

There are uncountably many ways to let time pass:

$$\text{if } q \xrightarrow{\delta} q' \text{ and } q' \xrightarrow{\epsilon} q'' \text{ then } q \xrightarrow{\delta+\epsilon} q''$$

## Time determinism property

There is exactly one state reached after a given delay:

$$|\{s' \mid s \xrightarrow{d} s'\}| = 1$$

# COMPLEX TIMED AUTOMATA

## Product Construction

We can define a product construction for timed automata so that a complex system can be defined as a product of component systems

Let  $A_1 = \langle L_1, L_{1_{start}}, \Sigma_1, X_1, I_1, F_1, E_1 \rangle$  and  $A_2 = \langle L_2, L_{2_{start}}, \Sigma_2, X_2, I_2, F_2, E_2 \rangle$  be two timed automata.

Assume that the clock sets  $X_1$  and  $X_2$  are disjoint, then, the **product automaton** is given as

$$A_1 \parallel A_2 = \langle L_1 \times L_2, L_{1_{start}} \times L_{2_{start}}, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, I, F_1 \times F_2, E \rangle$$

Where for locations  $\mathbf{s}_1$  and  $\mathbf{s}_2$  of  $\mathbf{A}_1$  and  $\mathbf{A}_2$  respectively, the **invariant**,  $I(s_1, s_2) = I(s_1) \wedge I(s_2)$ .

## Switches

The switches of a product automaton is defined by:

- for  $a \in \Sigma_1 \cap \Sigma_2$ , for every  $\langle s_1, a, g_1, r_1, s'_1 \rangle$  in  $E_1$  and  $\langle s_2, a, g_2, r_2, s'_2 \rangle$  in  $E_2$ ,  $E$  has  $\langle (s_1, s_2), a, g_1 \wedge g_2, r_1 \cup r_2, (s'_1, s'_2) \rangle$
- for  $a \in \Sigma_1 - \Sigma_2$ , for every  $\langle s, a, g, r, s' \rangle$  in  $E_1$  and every  $t$  in  $L_2$ ,  $E$  has  $\langle (s, t), a, g, r, (s', t) \rangle$
- for  $a \in \Sigma_2 - \Sigma_1$ , for every  $\langle s, a, g, r, s' \rangle$  in  $E_2$  and every  $t$  in  $L_1$ ,  $E$  has  $\langle (t, s), a, g, r, (t, s') \rangle$

Thus, locations of the product are pairs of **component-locations**, and the invariant of a compound location is the conjunction of the **invariants** of the **component locations**. The switches are obtained by synchronizing the switches with identical labels.

# REACHABILITY ANALYSIS

A location  $\mathbf{s}$  of the timed automaton  $\mathbf{A}$  is said to be reachable if some state  $\mathbf{q}$  with location component  $\mathbf{s}$  is a reachable state of the transition system  $\mathbf{S_A}$ . The input to the reachability problem consists of a timed automaton  $\mathbf{A}$  and a set  $F \subseteq L$  of *target* locations of  $\mathbf{A}$ .

The reachability problem is to determine whether or not some target location is reachable.

Since the transition system  $\mathbf{S_A}$  of a timed automaton is infinite, our solution to the reachability problem involves construction of finite quotients.

This is called a **time-abstract** transition system with transitions labeled only with symbols in  $\Sigma$  by hiding the labels denoting time increment.



# TIME ABSTRACT TRANSITION SYSTEMS

Let  $\mathbf{U}_A$  represent a time-abstract transition system obtained from a transition system  $\mathbf{S}_A$

- The state-space of  $\mathbf{U}_A$  equals the state-space  $\mathbf{Q}_A$  of  $\mathbf{S}_A$ .
- The set of initial states of  $\mathbf{U}_A$  equals the set of initial states of  $\mathbf{S}_A$
- The set of labels of  $\mathbf{U}_A$  equals the set  $\Sigma$  of labels of  $\mathbf{A}$ .

The transition relation of  $\mathbf{U}_A$  is the relation  $\Rightarrow$  such that;

for states  $q$  and  $q'$  and label  $a$ ,  $q \xRightarrow{a} q'$  iff there exists a state  $q''$  and a time value  $\delta \in \mathbb{R}^+$  such that  $q \xrightarrow{\delta} q'' \xrightarrow{a} q'$  holds in the transition system  $\mathbf{S}_A$

To determine reachability of target locations, we can consider the time-abstract transition system  $\mathbf{U}_A$  instead of the  $\mathbf{S}_A$  system

# STABLE QUOTIENTS

- Although the time-abstract transition system  $U_A$  has only finitely many labels, it still has infinitely many states.
- To address this problem, we consider equivalence relations,  $(\sim)$  over the state-space  $Q_A$

An equivalence relation,  $\sim$  over the state-space  $Q_A$  is said to be stable iff whenever  $q \sim u$  and  $q \xrightarrow{a} q'$ , there exists a state  $u'$  such that  $u \xrightarrow{a} u'$  and  $q' \sim u'$ .

The quotient of  $U_A$  with respect to a stable partition  $\sim$  is the transition system  $[U_A]_{\sim}$  such that states of  $[U_A]_{\sim}$  are the **equivalent classes** of  $\sim$ .

## Equivalence Class ( $\pi$ )

if for some  $q \in \pi$  and  $q' \in \pi'$ ,  $q \xrightarrow{a} q'$  holds in  $U_A$

To reduce the reachability problem  $(A, F)$  to a reachability problem over the quotient with respect to  $\sim$ , we need to ensure, apart from stability,  $\sim$  does not equate target states with non-target states.

An equivalence relation  $\sim$  is said to be  $F$ -sensitive, for a set  $F \subseteq L$  of target locations, if whenever  $(s, v) \sim (s', v')$ , either both  $s$  and  $s'$  belong to  $F$ , or both  $s$  and  $s'$  do not belong to  $F$ .

To solve the reachability problem  $(A, F)$ , we search for an equivalence relation  $\sim$  that is stable,  $F$ -sensitive, and has only finitely many equivalence classes.

We define an equivalence relation on the state-space of an automaton that equates two states with the same location if they agree on the integral parts of all clock values and on the ordering of the fractional parts of all clock values.

# REGION EQUIVALENCE

If two clocks  $x$  and  $y$  are between 0 and 1 in a state, then a transition with clock constraint ( $x = 1$ ) can be followed by a transition with clock constraint ( $y = 1$ ), depending on whether or not the current clock values satisfy ( $x < y$ ).

The integral parts of clock values can get arbitrarily large. But if a clock  $x$  is never compared with a constant greater than  $c$ , then its actual value, once it exceeds  $c$ , is of no consequence in deciding the allowed switches.

The integral parts of the clock values are needed to determine whether or not a particular clock constraint is met, whereas the ordering of the fractional parts is needed to decide which clock will change its integral part first

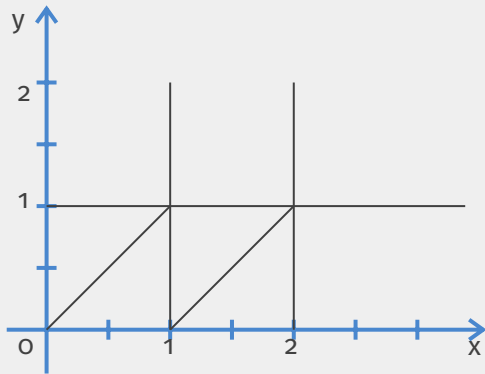
Given,  $\delta = \lfloor \delta \rfloor + fr(\delta)$  for any  $\delta \in \mathbb{R}^+$  where  $\lfloor \delta \rfloor$  is the integral part of  $\delta$  and  $fr(\delta)$  is the fractional part of  $\delta$ . For each clock  $x \in X$ , let  $c_x$  be the largest integer  $c$  such that  $x$  is compared with  $c$  in some clock constraint appearing in an invariant or a guard.

The region equivalence,  $\cong$  is defined over the set of all clock interpretations for  $X$ . For two clock interpretations  $v$  and  $v'$ ,  $v \cong v'$  iff all the following conditions hold:

- For all clocks  $x \in X$ , either  $\lfloor v(x) \rfloor$  and  $\lfloor v'(x) \rfloor$  are the same, or both  $v(x)$  and  $v'(x)$  exceed  $c_x$ .
- For all clocks  $x, y$  with  $v(x) \leq c_x$  and  $v(y) \leq c_y$ ,  $fr(v(x)) \leq fr(v(y))$  iff  $fr(v'(x)) \leq fr(v'(y))$ .
- For all clocks  $x \in X$  with  $v(x) \leq c_x$ ,  $fr(v(x)) = 0$  iff  $fr(v'(x)) = 0$

# CLOCK REGION

Consider a timed transition table with two clocks  $x$  and  $y$  with  $c_x = 2$  and  $c_y = 1$ .



# REGION AUTOMATON

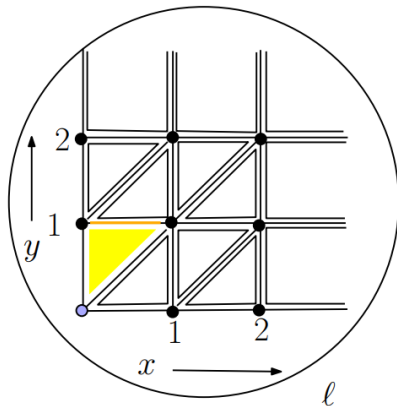
$(s, v) \cong (s', v')$  iff  $s = s'$  and  $v \cong v'$

The key property of region equivalence is its stability

The quotient  $[U_A]_{\cong}$  of a timed automaton with respect to the region equivalence is called the *region automaton* of  $A$ , and is denoted as  $R(A)$

The number of equivalence classes of  $\cong$  is finite, stable, and  $F$ -sensitive irrespective of the choice of the target locations. It follows that to solve the reachability problem  $(A, F)$ , we can search the finite region automaton  $R(A)$ .

# REGION GRAPH



$$x < 1, x > 0$$

$$y = 1$$

THIN

$$x < 1, x > 0$$

$$y < 1, y > 0$$

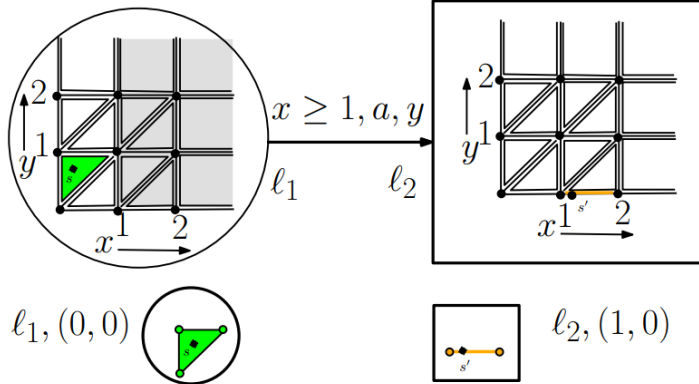
$$y - x > 0$$

THICK

$$x = 0, y = 0$$

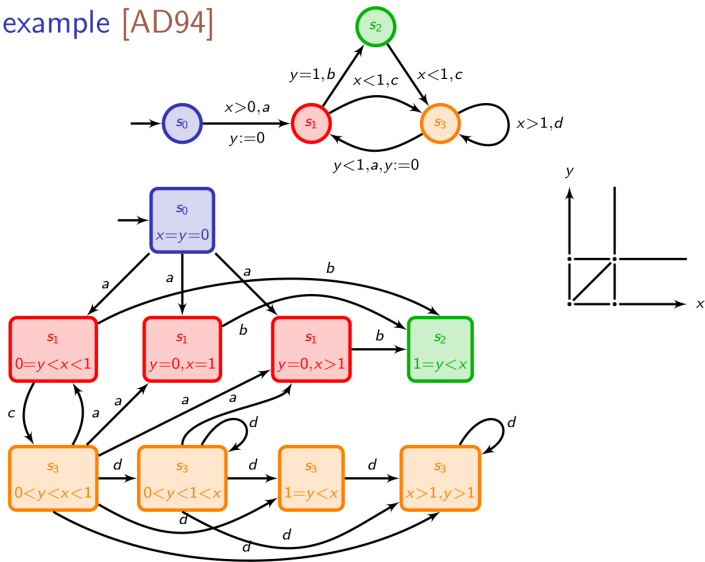
THIN





# DECIDABILITY

## An example [AD94]

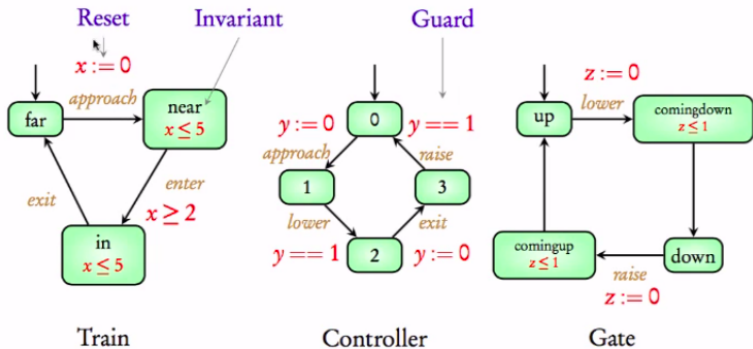


# DECIDABILITY ISSUES

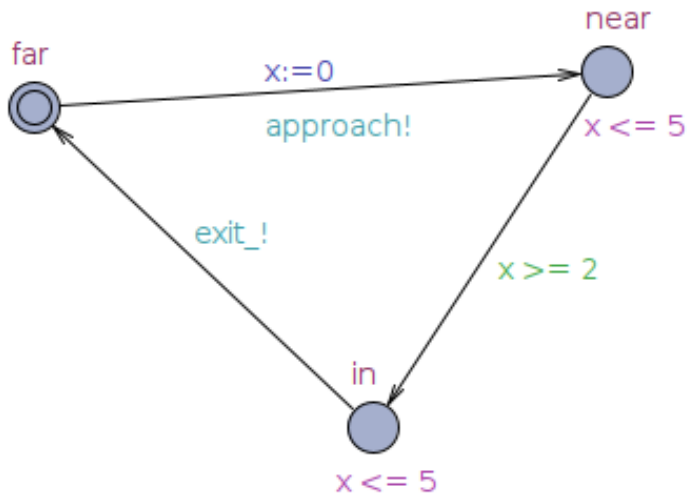
	Diagonal-free constraints	General constraints
$x := c, x := y$	PSPACE-complete	PSPACE-complete
$x := x + 1$		Undecidable
$x := y + c$		
$x := x - 1$	Undecidable	
$x < c$	PSPACE-complete	PSPACE-complete
$x > c$		Undecidable
$x \sim y + c$		
$y + c <: x < y + d$		
$y + c <: x < z + d$	Undecidable	

[Bouyer,Dufourd,Fleury,Petit 2000]

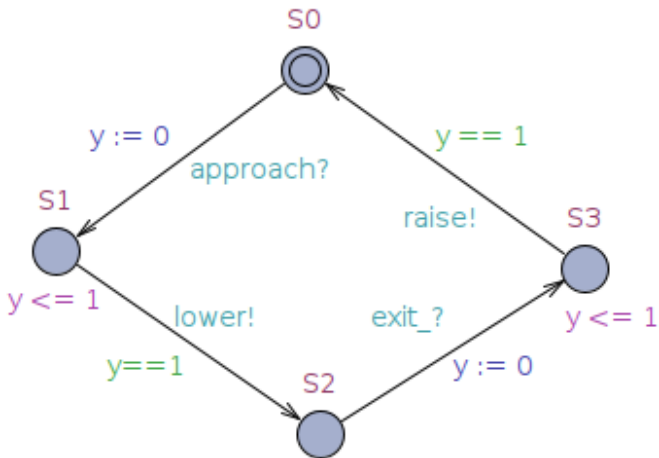
# UPPAAL IMPLEMENTATION



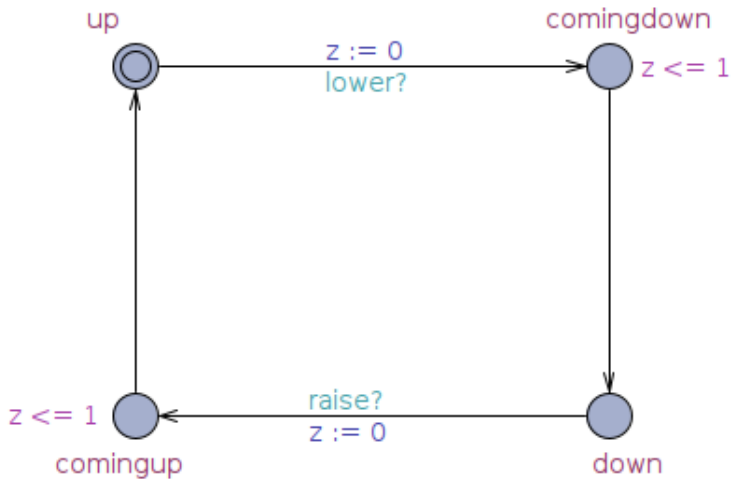
# TRAIN IMPLEMENTATION



# CONTROLLER IMPLEMENTATION

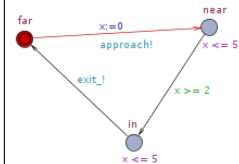


# GATE IMPLEMENTATION

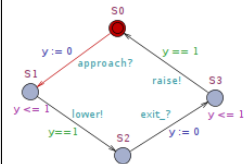


# RUN: STAGE 1 & 2

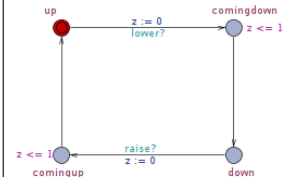
Train



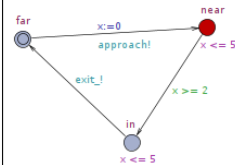
Controller



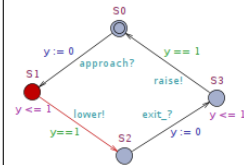
Gate



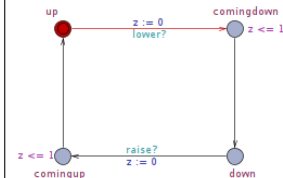
Train



Controller



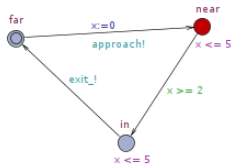
Gate



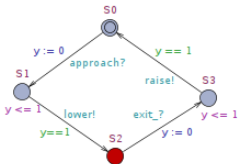


# RUN: STAGE 3 & 4

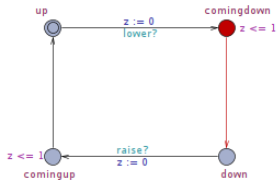
Train



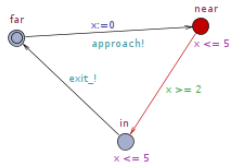
Controller



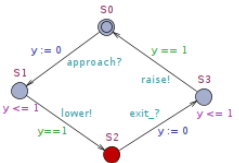
Gate



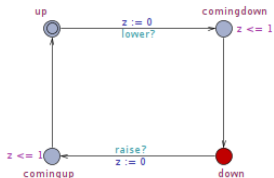
Train



Controller

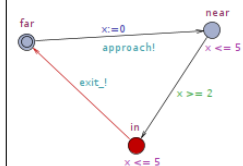


Gate

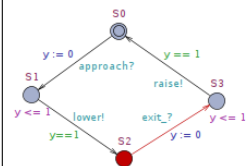


# RUN: STAGE 5 & 6

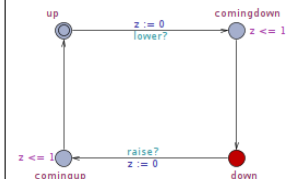
Train



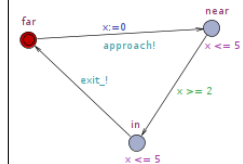
Controller



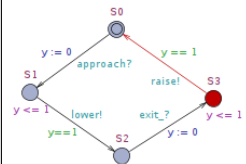
Gate



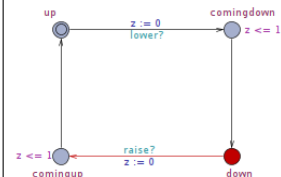
Train



Controller

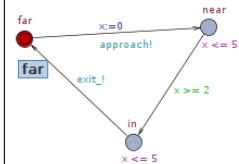


Gate

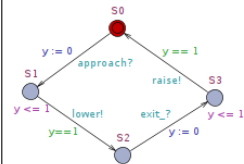


# RUN: STAGE 7 & 8

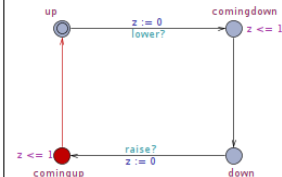
Train



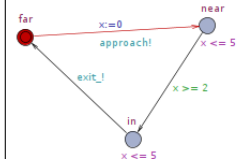
Controller



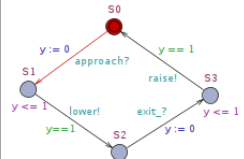
Gate



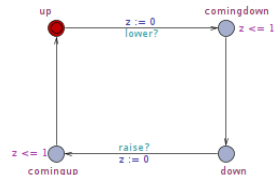
Train



Controller



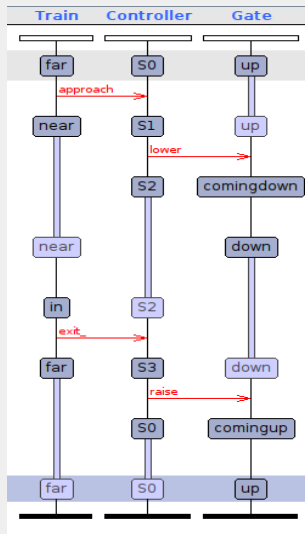
Gate



# SUMMARY

## Simulation Trace

(far, S0, up)  
approach: Train → Controller  
(near, S1, up)  
lower: Controller → Gate  
(near, S2, comingdown)  
Gate  
(near, S2, down)  
Train  
(in, S2, down)  
exit : Train → Controller  
(far, S3, down)  
raise: Controller → Gate  
(far, S0, comingup)  
Gate  
(far, S0, up)



# VERIFICATION

A[] (Train.in && Gate.up)



A[] !(Train.in && Gate.up)



A[] (Train.in && Gate.comingup)



A[] !(Train.in && Gate.comingup)



A[] (Train.in && Gate.comingdown)



A[] !(Train.in && Gate.comingdown)



# LIMITATIONS OF TIMED AUTOMATA

## Incomplete Specification

- Some delays may not be known yet, or may change

## Robustness

- What happens if time contain  $x = 5$  is implemented with  $x = 4.99$

## Optimization

- What is the impact of delay between two actions on the overall system

## Numerous Verification

- Timing delay changes in model requires a model check for the whole system






# CONCLUSION

# CONCLUSION

- Transition Systems
- Timed-automata
- Reachability analysis
- Decideability
- Implementation using UPPAAL
- Verification
- Challenges facing timed-automata



# REFERENCES

-  Rajeev Alur e David L. Dill. “A Theory of Timed Automata”. Em: *Theoretical Computer Science* 126 (1994), pp. 183–235.
-  Patricia Bouyer. *Timed Automata — From Theory to Implementation*. Invited tutorial, 6th Winter School on Modelling and Verifying Parallel Processes (MOVEP’04), Brussels, Belgium. 27 pages. Dez. de 2004.
-  Patricia Bouyer. *Timed Automata and Extensions: Decidability Limits*. Invited talk, 5èmes Journées Systèmes Infinis (JSI’05), Cachan, France. Mar. de 2005.
-  Rajeev Alur. *Timed Automata*. NATO-ASI 1998 Summer School on Verification of Digital and Hybrid Systems. A revised and shorter version appears in 11th International Conference on Computer-Aided Verification, LNCS 1633, pp. 8-22, Springer-Verlag, 1999.
-  *Timed automaton*.  
[https://en.wikipedia.org/wiki/Timed\\_automaton](https://en.wikipedia.org/wiki/Timed_automaton).  
Accessed: 2020-03-31.